Webová služba pro překlady/lokalizaci webových aplikací

Bc. Ondřej Žádník

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Žádník**
Osobní číslo:	**A15709**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Webová služba pro překlady/lokalizaci webových aplikací**
Téma anglicky:	**A Web Service for the Translation/Localisation of Web Applications**

Zásady pro vypracování:

1. **Nastudujte a srovnejte aktuálně používané metody pro překlady webových aplikací.**
2. **Shrňte funkční a nefunkční požadavky na webovou službu pro překlady aplikací.**
3. **Dle požadavků implementujte aplikaci pro překlady za využití některého z vývojových frameworků.**
4. **Navrhněte uživatelské rozhraní použitelné pro překladatele bez zkušeností s programováním.**
5. **Otestujte integraci vytvořené služby do již existující webové aplikace.**
6. **Demonstrujte proces překladů na webové aplikaci Hlidacky.cz před a po nasazení vyvinutého nástroje.**

Obr. 0.1

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **METZ, Sandi. Practical object-oriented design in Ruby: an agile primer. Upper Saddle River, NJ: Addison-Wesley, 2013. ISBN 0321721330.**
2. **RUBY, Sam, Dave THOMAS a DAVID HEINEMEIER HANSSON. Agile web development with Rails 4. Dallas, Texas: Pragmatic Bookshelf, 2013. ISBN 9781937785567.**
3. **HARTL, Michael. Ruby on Rails Tutorial: Learn Web Development with Rails. Addison-Wesley Professional, 2016, 816 s. ISBN 9780134597508.**
4. **FULTON, Hal. The Ruby way: solutions and techniques in Ruby programming. 3rd ed. Boston, Mass: Addison-Wesley, 2010. ISBN 0321714636.**
5. **HINZ, Yurek K. Exploring Open Source Software Localization Methods. Lambert Academic Publishing, 2011, 112 s. ISBN 9783844399738.**
6. **LEONARD RICHARDSON, Mike Amundsen. RESTful Web APIs. Sebastopol, Calif: O'Reilly, 2013, 816 s. ISBN 978-144-9359-737.**

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Obr. 0.2

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně .....

podpis autora

ABSTRAKT

Práce pojednává o aktuálně používaných metodách překladů webových aplikací. Nabízí shrnutí funkčních i nefunkčnich požadavků a srovnání jednotlivých řešení. Cílem bylo navrhnout webovou službu která umožní překladat webový obsah i lidem bez přístupu ke kódu programu. Služba by měla být schopná nahradit stavající řešení překladů na projektu Hlídačky.cz. Aplikace byla navržena v programovacím jazyce Ruby, frameworku Ruby on Rails.

Klíčová slova: translace, lokalizace, překlady, aplikace, ruby, ruby on rails

ABSTRACT

This thesis deals with actual translation methods of web applications. Offers a summary of functional and nonfunctional requirements and summary of possible solutions. The main target was to design web application that allow translating without access to a program code. Service should be able to replace current solution for translation on Hlidacky.cz project. Application was created in Ruby programming language with Ruby on rails web framework.

Keywords: translation, localization, globalization, ruby, ruby on rails

Rád bych poděkoval Ing. Radku Valovi za možnost zpracování takto zajímavého tématu. Mgr. Petru Šigutovi za odborné a přínosné rady. Velké díky patří také mojí rodině a přítelkyni za velkou trpělivost.

OBSAH

ÚVOD

Diplomová práce je zaměřena na navržení služby pro překlady webových aplikací a vysvětlení jednotlivých pojmů ohledně překladů. Zabývá se také popisem aktuálně používaných metod k překladu webových aplikací. Poskytuje souhrn jejich výhod a nevýhod. Podle nich byla navržena nová aplikace, která by měla umožnit překlad i lidem bez přístupu ke kódu aplikace. Uživatelské rozhraní by mělo být jednoduché, intuitivní a rychlé k používání.

Aplikace byla otestována na startup projektu Hlídačky.cz. Ten se jevil jako vhodný kandidát, pro otestování aplikace z důvodu příliš složitého procesu překládání textů v aplikaci. S ohledem na projekt Hlídačky.cz byl zvolen programovací jazyk Ruby a framework Ruby on Rails. Stejný jako na zmíněném projektu. Proces překladů by se měl za použití aplikace zjednodušit a zefektivnit využití času na projektu.

Práce by měla rozšířit obzory o možných metodách překladů a ukázat možnost jak navrhnout aplikaci pro překlady vlastních projektů.

I. TEORETICKÁ ČÁST

1 Motivace

V dnešní době, kdy po celém světě vzniká velká spousta IT startup projektů, začíná být vícejazyčnost téměř nutností. Firmy se snaží pronikat na zahraniční trhy a tím vzniká potřeba překladů co cizích jazyků. Myslím že potřeba kvalitních a bezchybných překladů je nesmírně důležitá pokud má na zákazníky působit věrohodně. Není nic horšího než kvalitní produkt, který je neprofesionálně přeložen.

Zde přichází na řadu úvaha- co má kvalita překladů společného s metodami překladů? Problém je v tom, že překlady většinou v aplikaci probíhají na úrovni zásahu do zdrojového kódu. Tím pádem je nereálné, aby se k překladům dostaly externí překladatelé. Pokud už se k tomu dostanou, je jím posílán jakýsi balík toho co je potřeba přeložit a po přeložení jej musí odeslat zpět a programátor musí tento balík zkontrolovat, zda neporušil strukturu nezbytnou pro načtení a poté jej nahrát zpět do aplikace.Proces je tedy velice zdlouhavý a časově náročný. Zejména časová náročnost je u startup projektů velice důležitá.

Proto se většina těchto projektů uchyluje k rychlejší variantě a to je překládání již zmíněných balíků programátory. Odpadá tedy proces odesílání a přijímaní. Pojem programátor je, ale už ze své podstaty něčím jiným než překladatelem. Mohou tedy vznikat překlady které neodpovídají danému kontextu, projektu či business logice.

2 Překlady obecně

S celým celým tématem lokalizace software či webové aplikace souvisí několik velice důležitých pojmů. Jde o lokalizaci, globalizaci a internationalizaci. Všechny tyto pojmy jsou základním stavebním kamenem překladů webových aplikací.

2.1 Překlad

Proces, při kterém překladatel převádí text z jednoho jazyka do jiného. Je potřeba dbát na jazykové i významové aspekty textu. Důležitými parametry jsou stylistika, gramatika, terminologie, rozložení textu a délka textu.

2.2 Internacionalizace

Chápeme jí jako takovou přípravu před lokalizací samotnou. Po jejím provedení je program, který dříve dokázal pracovat s jedním jazykem, schopen pracovat ve více jazycích. Programy užívající řetězce v jednom jazyce, jsou spojeny do obecných cest. Ty jsou dále využívány pro hledání překladů pro jiné jazyky.

2.3 Lokalizace

Lokalizace v již internacionalizovaných programech poskytuje informace které mohou převést vstupy a výstupy do tvaru pro daný jazyk. Lokalizací rozumíme překladatelskou část. Můžeme ji vyjádřit jako změnu celého produkt tak, aby vyhovoval cílové oblasti z jazykového, kulturního, obchodního a technického hlediska. Docházet může i ke změně struktury projektu.

2.4 Globaliza

Globalizace popisuje celý proces vývoje a adaptování produktu k jeho užití ve více zemích. Lze ji také chápat jako kombinaci internacionalizace a jednotlivých lokalizací. Při globalizaci je zároveň nutné vzít v potaz rozdílný trh a marketingovou strategii.

2.5 Locale

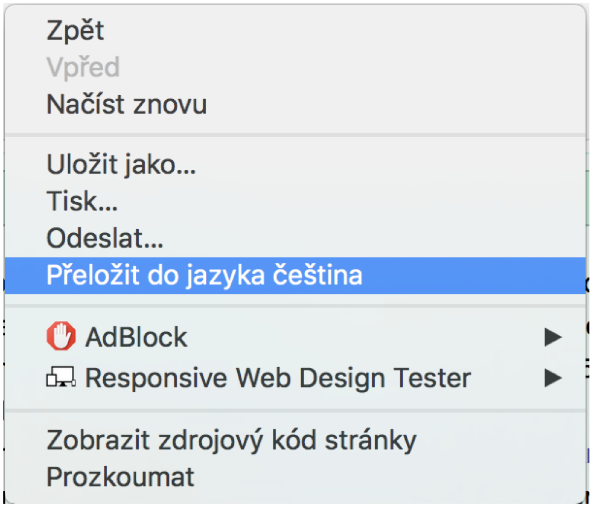
Termín označuje aktuální jazykové prostředí, které se vztahuje k vybrané zemi. Zkratka představuje označení jazyka a země s ním spojené. Pokud tedy země jako Česká republika má hlavní jazyk češtinu, je locale cs_CZ. První parametr před lomítkem označuje jazyk a druhý zemi.

3 Překlady webových aplikací

Popíšu zde aktuálně používané metody pro překlady webových aplikací. Kvůli ručním úpravám překladů bylo vyvinuto nemálo služeb či aplikací, které se pokoušejí tento proces co nejvíce zjednodušit. Já se zde pokusím vyzdvihnout jen ty, o kterých si myslím že mají zajímavý přístup k této problematice.

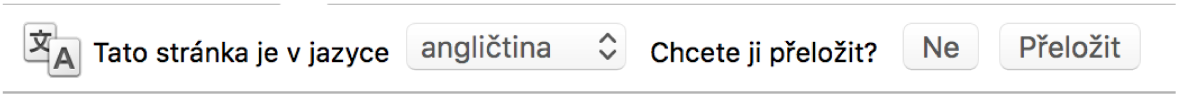
3.1 Google - Automatický překlad stránky

Google implementoval do svého prohlížeče Chrome možnost přeložit stránku z rozpoznaného jazyka, do aktulního jazyka prohlížeče. Stačí jít na stránku psanou v jiném jazyce než výchozím a google nabídne možnost překladu. Popř. se volba zobrazí po stisknutím pravého tlačítka myši.

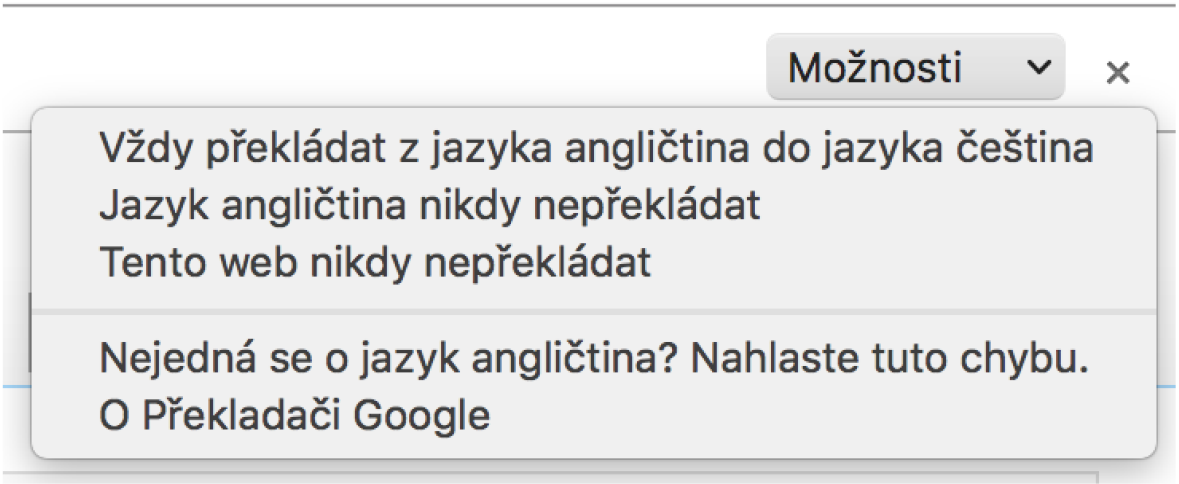


Obr. 3.1 Volba pro přeložení webové stránky službou Google

Po zvolení je stránka přeložena a zobrazena, původní stránku je možno zobrazit také. Nabídnuta je také možnost ručně přeložit cokoliv dalšího.



Obr. 3.2 Nabídka přeložení webové stránky při identifikaci cizího jazyka



Obr. 3.3 Možnosti přeložení webové stránky

Bohužel tento překlad je pouze strojový a ne přílíš přesný. Dá se ovšem použít jako nouzové řešení, případně nápověda pro neznámá slovíčka. Google se o tento produkt ale dobře stará a postupem času překlady vylepšuje.

3.2 Nástroje pro lokalizaci

Lokalizační software, jehož funkcí je extrakce textových řětezců do externí aplikace, kde dojde k jejich přeložení do požadovaných jazyků. Následně vygeneruje nové soubory s překlady které jsou importovány do aplikace a ta je schopna z nich načítat přeložené textové řetězce.

Lokalizační software by měl poskytnout jednoduché a přehledné uživatelské rozhraní. Jasně by měl definovat, pro jaké jazyky je možné překládat a o jaký překlad se jedná, popř. kde se překlad na stránce nachází.

Samozřejmostí by měl být překlad řetězců s využitím nějakých proměnných. Můžeme potom více využít dynamiku a variabilitu těchto překladů. Překládání potom nebude čistě jen pro statické texty.

Většina aplikací tohoto typu je ovšem zpoplatněna, za vyšší cenu často nabízejí premiové funkce jako je například našeptávání z podobných překladů, napovídání z externího překladače a spousty dalších.

3.2.1 LocaleApp

LocaleApp je nástroj pro práci s překlady ve formátu YAML. Ten je používán ve webovém frameworku Ruby on Rails, napsaný v jazyce Ruby. Poskytuje nástroj pro import a export překladů. Poskytuje možnost více překladatelů, přehled přeložených a nepřeložených textů, vyhledávání v překladech.

Do 500 překladů a 2 jazyků zdarma, poté placený. Nejlevnější tarif začíná na 24 eurech za měsíc a končí na téměř 400 eurech za měsíc.

Výhodou je jednoduchá propojitelnost s frameworky používající YAML klíče. Nevýhodou je již zmíněný YAML formát. Programátor je nucen místo základního překladu psát klíče které symbolizují jakýsi základní text, který je poté přeložen.

3.2.2 Translation IO

Translation IO je online nástroj využívající knihovnu GNU GetText. GetText jako takový je multiplatformní nástroj. Translation IO je opět pro Ruby on Rails framework. Jeho výhodou je jednoduchý import projektů používajících YAML. Jako další přednost bych vyzdvihl možnost použít zároveň GetText i YAML. Jeho jednoduchou a rychlou synchronizaci. Bohužel uživatelské rozhraní je chaotické a často nepřehledné, vyhledávání a řazení je taktéž nepříliš intuitivní.

Ceník se narozdíl od LocaleApp odvíjí od počtu klíčů uložešných v aplikaci. Velkou výhodou která plyne přímo z GetTextu, je možnost psát rovnou texty v základním jazyce. Ty jsou potom uloženy do Translation IO aplikace a tam následně přeloženy.

3.2.3 LocalizeJS

LoacalizeJS je zajímavý moderní nástroj pro překlady. Dokáže překládat webové aplikace, mobilní aplikace, souborové systémy. Jeho zajímavostí je, že stačí vložit krátký javascriptový kód, který načte LocalizeJS do naší aplikace. Všechny překlady jsou potom přeloženy v LocalizeJS rozhraní. Javascript poté zastíní originální texty a před zobrazením uživateli zobrazí přeložený text. Tohle řešení je hodně rozdílné, na jednu stranu moderní a efektivní řešení. Na druhou stranu přináší problémy spojené s javscriptem. Taktéž způsobuje problémy se SEO.

II. PROJEKTOVÁ ČÁST

4 Požadavky na aplikaci

Startup Hlídačky.cz momentálně používá aplikaci Translation IO. Mezi hlavními požadavky bylo snížení nákladů, proto padla volba na vlastní aplikaci. Neméně důležitým požadavkem je zjednodušení uživatelského rozhraní a možnost překladů bez nutnosti zásahu do zdrojového kódu. Aplikace by měla být intuitivní a mít jednoduché ovládací prvky.

Synchronizování nově přeložených textů s aplikací Hlídačky.cz by nemělo být přílíš zdlouhavé ani složité. Aplikace by měla umožnit přidání více překladatelů nebo možnost vytvořit více projektů pro přeložení. Integrace nově vytvořené aplikace by měla být jednoduchá a nabídnout možnosti výběru překládaných jazyků.

Stávající překlady ve formátu GetText musí být zachovány a již přeložené texty uloženy do nového systému překladů. Aplikace bude rozdělena na client-server řešení. Řešení clienta je integrovatelné do projektu, který se chystáme překládat. Serverová část má za úkol postarat se o přijímání textů, následný překlad a zpětnou synchronizaci do aplikace.

5 Použité technologie

Pro potřeby pplikace byl zvolen programovací jazyk Ruby a framework pro webové aplikace Ruby on Rails. Jako datová základna byla použita databáze PostgreSQL. Pro zachování stávajících překladů a uložení nových byla zvolena knihovna GNU GetText. Pro vytvoření uživatelského rozhraní byl použit framework kaskádových stylů Bootstrap 3.

5.1 GIT

Git je distribuovaný systém pro verzování. Vytvořen byl Linusem Torvaldsem pro správu verzí Linuxového jádra. Poté se rozšířil i na jiné projekty. Jádro projektu Git se rozrostlo na kompletní systém, který je použitelný bez doplňkových nástrojů.

- Nelineární vývoj. Git podporuje rychlé práci ve více větvích a jejich rychlé slučování.
- Specifické nástroje pro vizualizaci a navigaci v historii vývoje projektu.
- Distribuovaný vývoj. Podobný jako BitKeeper, Mercurial, SVK.
- Poskytuje každému vývojáři lokální kopii historie vývoje.
- Změny jsou přenášeny z jednoho úložiště do jiného.

5.2 GitHub

GitHub je online služba, která umožňuje ukládat GIT data na server. Jsou proto dostupná odkudkoliv. Je možno vytvořit více repozitářů přes webové rozhraní. Každý repozitář umožňuje správu uživatelů a spolupracovníků. Přehledne zobrazuje historii změň kódu s informacemi kdo a kdy změnu provedl.

GitHub je možné provozovat ve dvou verzích. Placena a neplacená. Neplacená verze nabídne pouze veřejné repozitáře, kdežto verze placená nabídne dle zaplaceného tarifu omezený počet privátních repozitářů. Alternativou GitHubu může být Bitbucket, který je obdobný, ale nabízí zdarma i soukromé repozitáře.

Já jsem pro projekt zvolil GitHub, protože mi jeho uživatelské rozhraní přišlo přívětivější a příjemnější.

- Nelineární vývoj. Git podporuje rychlé práci ve více větvích a jejich rychlé slučování.
- Specifické nástroje pro vizualizaci a navigaci v historii vývoje projektu.

- Distribuovaný vývoj. Podobný jako BitKeeper, Mercurial, SVK.
- Poskytuje každému vývojáři lokální kopii historie vývoje.
- Změny jsou přenášeny z jednoho takového úložiště do jiného.

5.3 Ruby

Interpretovaný skriptovací programovací jazyk. Jeho syntax je jednoduchá a intuitivní k naučení. Je plně objektový, to značí že všechno s čím se v jazyce Ruby setkáme je objekt. Řadí se k ostatní novodobým programovacím jazykům jako je Python nebo Perl. Je multiplatformní a dynamicky typovaný. Oblíben je zejména díky výbornému frameworku pro webové aplikace Ruby on Rails.

Výhody:

- interpretovaný jazyk tzn. není nutná kompilace kódu
- multiplatformnost
- IRB režim
- jednoduchost syntaxe
- plně objektově orientovaný
- dynamicky typovaný
- možnost tvorby GUI

Nevýhody:

- oproti jazykům kompilovaným je pomalejší(překlad za běhu)

5.4 Ruby on Rails

RoR zastává názor, že uživatel konfiguruje jen ty části, které jsou nezbytně nutné popř. se liší od jeho základního principu. Mnoho programátorů může tento přístup odradit, protože nutí uživatele do jeho "nejlepšího postupu". Pokud si programátor navykne na tento přístup, může tím mnohonásobně zvýšit svoji produktivitu.

5.4.1 MVC

Ruby on Rails je založený na MVC architektuře. Ta značí architekturu Model, View, Controller. 3 základní vrstvy programu. Modelová vrstva zapouzdřuje aplikační logiku a metody pro práci s databází. View je pohledová vrstva, která se stará o zobrazování dat. Controller je prostředníkem mezi modelovou a pohledovou vrstvou.

Výhody MVC:

- Separování aplikační logiky a uživatelského rozhraní
- Znovupoužitelnost kódu
- Přehlednější struktura kódu

5.4.2 REST

Rest je architektura rozhraní. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům. Zdrojem jsou data a stavy aplikace. Všechny zdroje mají identifikátor URI a REST definuje čtyři základní metody pro přístup k nim.

Metody pro přístup ke zdrojům: REST poskytuje čtyři základní metody. Těmto základním 4 metodám říkáme CRUD(Create - Vytvoření, Retrieve - Získání, Update - Upravit, Destroy - Smazat). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.

- GET(Retrieve) - Metoda pro získání zdroje
- POST(Create) - Metoda pro vytvoření dat, většinou HTML formuláře
- PUT(Update) - Metoda pro smazání zdroje
- DELETE(Delete) - Metoda pro úpravu zdroje

5.4.3 Modely

Model značí data v aplikaci a pravidla pro práci s nimi. V Ruby on Rails je model určen pro práci s databázovou tabulkou a pro metody určené k těmto datům. Většinou jeden model odpovídá jedné databázové tabulce. Modely by měly obsahovat nejvíce aplikační logiky.

5.4.4 Kontrollery

Kontrollery můžeme definovat jako nějaký spoj mezi modely a pohledy. Slouží ke zpracování požadavků z webového prohlížeče, získávání dat z modelů a předávání těchto dat do pohledů, kde jsou zobrazeny uživatelům.

5.4.5 Pohledy

Pohledy představují uživatelské rozhraní aplikace. Obvykle prezentované jako HTML soubory s příměsí ruby kódu. Ruby kód by měl sloužit pouze k reprezentaci dat, nikoliv k výpočtům či nějaké složitější logice.

Ruby on Rails obsahuje více částí:

- Action Pack
- Action Controller
- Action Dispatch
- Action View
- Action Mailer
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

5.4.6 Action Pack

Gem, který se stará o kontrolery a pohledy. Obsauje třídy jako Action Controller, Action Dispatch a Action View.

5.4.7 Action Controller

Balík, který se stará o kontrolery. Zpracovává příchozí požadavky s jejich parametry a poté je přesměrovává na příslušnou akci, která v sobě zapouzdřuje logiku. Poskytuje také sessions, logiku pro renderování šablon a směrování.

5.4.8 Action view

Balík starající se o pohledy aplikace. Nabízí HTML a XML výstup, řídí renderování šablon, stará se o vnořené šablony a parcialní šablony. Obsahuje speciální funkce pro práci s Ajaxem.

5.4.9 Action Dispatch

Stará se o směrování příchozích požadavků.

5.4.10 Action Mailer

Kompletní balík pro práci s e-maily. Obsahuje veškerou logiku na rozesílání a příjem e-mailů a jejich šablony.

5.4.11 Active Model

Balík, který definuje rozhraní pro práci se službami Action Packu a vrstvami ORM jako je např. Active Record či MongoDB. Poskytuje možnost práce s jinýma technologie pro práci s databázemi.

5.4.12 Active Record

Balík Active Record je základním prvkem pro práci s modely. Odstíňuje programátora od čistého SQL a poskytuje mu metody nezávislé na zvoleném SQL jazyku. Poskytuje metody pro vytváření, editace, mazání a získávání dat z databáze. Dále složitější operace jako je JOIN či GROUP. Metody pro řazení a definování vztahů mezi modely.

Active Record je základem pro modely v Rails aplikaci. Poskytuje mimo jiné nezávislost na SQL dialektu konkrétní databáze, základní operace pro získávání, vytváření, editaci a mazání záznamů (tzv. CRUD, Create Read Update Delete), pokročilé získávání dat z databáze (operace typu JOIN, GROUP, aj.) a v neposlední řadě i možnost definovat vztahy mezi jednotlivými modely.

5.4.13 Active Resource

Active Resource umožňuje propojení modelů s webovými REST službami.

5.4.14 Active Support

Balík, který přidává velké množství rozšíření pro základní Ruby tříd.

5.4.15 Railties

Railties umožňují propojení všech jednotlivých součástí dohromady.

Principy Ruby on Rails:

- DRY — „Don't repeat yourself“ aneb „neopakujte se“ — vyvarování se psaní stejného kódu na více místech, nahraďte jej znovupoužitelným kódem

- Konvence před konfigurací — Rails mají sadu pravidel podle kterých konfigurují nastavení frameworku
- REST návrhový vzor — Webová aplikace jako soubor „zdrojů“ (resources) a standardních HTTP „sloves“ (verbs) je nejsnazší a nejrychlejší způsob, jak modelovat entity a jejich vztahy

Části ze kterých se skládá Ruby on Rails:

- DRY — „Don’t repeat yourself“ aneb „neopakujte se“ — vyvarování se psaní stejného kódu na více místech, nahraďte jej znovupoužitelným kódem
- Konvence před konfigurací — Rails mají sadu pravidel podle kterých konfigurují nastavení frameworku
- REST návrhový vzor — Webová aplikace jako soubor „zdrojů“ (resources) a standardních HTTP „sloves“ (verbs) je nejsnazší a nejrychlejší způsob, jak modelovat entity a jejich vztahy

5.4.16 Gemy

Rails aplikace používá gemy, ty pracují pod nástrojem zvaný Bundler. Při vygenerování nové aplikace, Rails vytvoří soubor s názvem Gemfile, kde jsou specifikované všechny gemy používané v aplikaci.

5.4.17 RubyGems

Balíčkovací systém pro Ruby. Balíčky jsou umístěny na http://rubygems.org/

5.4.18 Bundler

Nádstavba nad RubyGems, spravuje závislosti pro jednotlivé knihovny. Gemfile drží informaci o použitých knihovnách. Po instalaci se do souboru Gemfile.lock zapíšou konkrétní verze a vazby gemů.

5.4.19 Rake

Rake jde po vzoru klasického make, ale s tím rozdílm že je pro jazyk Ruby. Dokáže spoustět úlohy z příkazového řádky i z kódu. Nejtypičtějším příkladem mohou být migrace či nějaké pomocné úkoly.

Soubor/Složka	Účel
Gemfile	Specifikace gemů (a případně jejich verzí), které vaše aplikace vyžaduje.
README.rdoc	Návod k aplikaci. Zde popište, co vaše aplikace dělá, jak ji nainstalovat, nastavit, apod. Slouží též jako úvodní stránka vygenerované dokumentace (viz příkaz `rake doc:app`).
Rakefile	Natahuje definice konzolových příkazů pro nástroj Rake (definované v Rails a ve vaší aplikaci).
app/	Obsahuje kontrolery (*controllers_), modely (models_) a pohledy (views*).
config/	Konfigurace aplikace, směrování požadavků (`routes.rb`), definice přístupových údajů k databázi a podobně.
config.ru	Konfigurace pro spuštění aplikace prostřednictvím webových serverů využívajících rozhraní Rack.
db/	Aktuální schéma databáze a databázové migrace.
doc/	Podrobná dokumentace pro vaši aplikaci.
lib/	Rozšíření a doplňkové soubory vyžadované vaší aplikací.
log/	Logy aplikace.
public/	Jediná složka, která je přístupná přes web. Obsahuje obrázky, JavaScript, stylové předpisy (CSS) a další statické soubory.
script/	Obsahuje skript `rails`, který spouští vaši aplikaci ve vývojovém prostředí, generuje kód, spouští interaktivní konzoli aplikace, a podobně.
test/	Jednotkové a funkční testy, testovací data a ostatní nástroje pro automatizované testování aplikace. Ty jsou popsány v návodu Testing Rails Applications
tmp/	Dočasné soubory
vendor/	Místo pro zdrojové soubory třetích stran. V typické Rails aplikaci jsou to například gemy a pluginy rozšiřující funkce Rails, případně též zdrojový kód Rails.

Obr. 5.1 Základní struktura Rails aplikace

5.5 PostgreSQL

Relační databázový systém s open-source licencí. Výborná spolehlivost, bezpečnost a výkonnost. Má podporu většiny operačních systémů. Nabízí podporu cizích klíčů, JOIN operací, pohledů a transakcí. Obsahuje většinu základních SQL datových typů a struktur. V novějších verzích nabízí moderní datové typy jako ARRAY, HSTORE, JSON, JSONB.

5.6 YAML formát

Formát pro serializaci strukturovaných dat. Nabízí jednoduchost, čitelnost a přehlednost bez nutnosti psaní složitých struktur či mnohonásobného uzávorkování. YAML používá čistý text k reprezentaci dat, mezery a odřádkování k tvoření struktur. Poměrně jednoduchým způsobem dokážeme zpracovat i složitější strukturu. S tímto formátem se můžeme setkat nejčastěji u vývojářů Ruby, knihovny pro ostatní jazyky jsou ovšem také dostupné. .

- Velice dobrá čitelnost člověkem i strojem
- Struktury tvořeny pouze odsazováním, pomocí 2 či 4 mezer
- Neomezené úrovně zanoření

```yaml
# An employee record
martin:
    name: Martin D'vloper
    job: Developer
    skill: Elite
```

Obr. 5.2 Ukázka YAML struktury

5.7 GetText

Lokalizační sada nástrojů spadající pod projekt GNU, který umožňuje tvorbu vícejazyčných aplikací. Ze zdrojových kódů jsou extrahovány překlady, které začínají specifickým prefixem _(. Z těch je vytvořena šablona formátu .pot. Z těchto šablon jsou později vytvořeny tzv. katalogy ve formátu .po které nesou klíč a překlad v daném jazyce.

GetText je rozšířen napříč jazyky:

- C (C, C++, Objective C)

- Python
- Ruby
- PHP
- Java
- a další.

```
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: OLPC website files Version 1.2\n"
"POT-Creation-Date: 2006-02-14 12:00-0500\n"
"PO-Revision-Date: 2006-08-27 15:00-0500\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"

msgid "align"
msgstr "_: the alignment (left or right) of your language goes here"

#: laptop.html
#. An instruction to translators
msgid "title"
msgstr "One Laptop per Child"

msgid "intro"
msgstr "Introducing the children's laptop..."
```

Obr. 5.3 Ukázka POT souboru

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR VideoLAN
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: vlc-devel@videolan.org\n"
"POT-Creation-Date: 2009-03-26 21:58+0000\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"X-Generator: Translate Toolkit 1.3.0\n"

#: include/vlc_common.h:879
msgid ""
"This program comes with NO WARRANTY, to the extent permitted by law.\n"
"You may redistribute it under the terms of the GNU General Public License;\n"
"see the file named COPYING for details.\n"
"Written by the VideoLAN team; see the AUTHORS file.\n"
msgstr ""
"Ce programme est fourni SANS AUCUNE GARANTIE, tel que permis par la loi.\n"
"Vous pouvez le redistribuer selon les termes de la Licence Publique GÃ©nÃ©rale "
"GNU ;\n"
"voir le fichier COPYING pour plus de dÃ©tails.\n"
"Ã‰crit par lâ€™Ã©quipe VideoLAN ; voir le fichier AUTHORS.\n"

#: include/vlc_config_cat.h:32
msgid "VLC preferences"
msgstr "Préférences de VLC"
```

Obr. 5.4 Ukázka PO souboru

5.8 CSS a Sass

CSS neboli kaskádové styly je zápis, který určuje vzhled HTML dokumentu. Soubory s kaskádovými styly používají příponu css. Soubory kaskádových stylů jsou úzce provázány s HTML soubory.

Sass je kompilovaný jazyk a je jakousi nástavbou klasického CSS které obohacuje o proměnné, cykly, podmínky, mixiny, funkce aj. Jeho syntax poskytuje možnost dědičnosti a dle jednoduchého zanořování umožňuje zmenšení CSS souborů.

5.9 Bootstrap 3

Bootstrap 3 od Twitteru je skvělý CSS framework, který ulehčuje tvorbu čehokoliv responsivního a pomocí spousty pomocných tříd a modulů výrazně urychluje práci s designem. Poskytuje navíc moduly pro práci s mobilním designem.

5.10 Font awesome

Font Awesome poskytuje sadu ikon pro použití na webu. U ikon můžeme upravovat velikost, barvu, stín, většinu atributů co umí CSS.Font Awesome je šířen pod licencí Open Font License (OFL).

Celá sada ikon je ve své podstatě písmo a můžeme na něj tedy aplikovat kaskádové styly pro práci s písmem. Lze libovolně manipulovat s velikostí, barvami, tučností. Při zvětšení ikony uchovávají svoji kvalitu protože nejsou obrázkem.

5.11 jQuery

jQuery je javascriptová knihovna, která má podporu většiny moderních prohlížečů. Klade důraz na spolupráci mezi JavaScriptem a HTML. jQuery odděluje „chování“ od struktury HTML. Místo příme specifikace eventu v HTML kódu, je funkcionalita oddělena do samostatného souboru, kde je specifikováno jaký element má provádět jakou akci.

jQuery nabízí následující funkce:

- Výběr DOM
- Funkce pro procházení a změnu DOM
- Události, efekty a animace
- Editace CSS
- AJAX
- Rozšiřitelnost
- Javascriptové pluginy

5.12 Heroku

Heroku je cloudová hosting platforma jako služba (PaaS), která nabízí podporu mnoha programovacích jazyků. Heroku pro vývoj aplikací využívá git a pro jeho manipulaci je tak třeba příkazový řádek nebo webová stránka samotná. Heroku má vlastního CLI klienta, kterého si můžete stáhnout na většinu tradičních os. Po instalaci heroku klienta, máme k dispozici většinu heroku příkazů.

Heroku se od běžného FTP hostingu liší v tom, že k němu přistupujeme přes GIT. Heroku je znám svou velkou kvalitou poskytovaných služeb, která je ovšem vykoupena vyšší cenou. Určitě bych jej doporučil k vyzkoušení.

5.13 Base64

Base64 je kódování. Převádí binární data do množiny tisknutelných znaků. Umožňuje přenos binárních dat tam, kde je povolen pouze přenos textových řetězců. Nejčastější použití je přenos MIME souborů.

Toto kódování se nejčastěji používá v některých aplikačních protokolech.Tam kde potřebujeme aby přenášená binární data, nenarušila konvence aplikačního protokolu a neznemožnila přenos informace. Kódování Base64 například najdeme ve standardu MIME, který používáme v e-mailech, abychom odeslali text s diakritikou nebo přílohu. Kódování Base64 můžeme použít i k přenosu binárních souborů.

5.14 JSON

Formát pro výměnu dat. Jednoduše čitelný i zapisovatelný člověkem. Snadno rozpoznatelný i generovatelný strojově. Je založen na jazyku JavaScript. JSON je reprezentován v textové podobě, nezávislý na zvoleném programovacím jazyku. Díky tomu je JSON pro výměnu dat opravdu ideálním jazykem

JSON používá dvě struktury:

- Kolekce párů název/hodnota. Ta bývá v rozličných jazycích realizována jako objekt, záznam (record), struktura (struct), slovník (dictionary), hash tabulka, klíčový seznam (keyed list) nebo asociativní pole.
- Seřazený seznam hodnot. Ten je ve většině jazyků realizován jako pole, vektor, seznam (list) nebo posloupnost (sequence).

6 Implementace aplikace Klient-Server

Aplikace je rozdělena na dvě části. Klientská část byla pojmenována Condor a serverová část byla pojmenována Condor-app. Dále se tedy na ně budu odkazovat pod těmito označeními.

6.1 Klientská část - Condor

První část je ve formě Ruby on Rails gemu, který se přidá do stávající aplikace. Co je to ruby gem už jsem zmínil výše. Zde ho rozeberu více do hloubky. Ruby gem nemá k dispozici tolik pomocných tříd a metod jako Ruby on Rails aplikace. Je to jen soubor základních tříd s jednoduchou konfigurací.

Soubor/Složka	Účel
bin/	Složka se základním nastavením, načítá základní části
lib/	Složka obsahující uživatelem definované třídy pro práci gemu
test/	Složka obsahuje veškeré soubory a pomocné třídy k testovacímu prostředí Rake
gem.gemspec	Konfigurační soubor gemu, obsahuje základní nastavení a nastavení testovacího prostředí gemu a závislosti na ostatních gemech
Rakefile	Soubor pro definice úloh pro nástroj Rake

Obr. 6.1 Struktura ruby gemu

6.1.1 header třídy condor.rb

Listing 1 condor.rb

```
# -*- encoding: utf-8 -*-
require 'fast_gettext'
require 'language_list'
require 'yaml'
require 'gettext/tools'
require 'net/http'
require 'json'
require 'base64'
```

6.1.2 require

Require je speciální příkaz, který říká ruby kódu jaké knihovny má natáhnout. Ruby gem v základu nemá žádné speciální pomocné vazby, je tedy potřeba všechny specifické knihovny které k práci potřebujeme, definovat explicitně.

6.1.3 FastGettext

Gem postavený nad knihovnou GNU GetText. Přidává spoustu pomocných metod a funkcí pro práci s GetTextem.

6.1.4 LanguageList

Gem který přidává seznam všech jazyků v normě ISO-639-1 a ISO-639-3. Přidává metody pro vyhledávání a práci s nimi.

6.1.5 YAML

Ruby knihovna pro práci s YAML soubory.

6.1.6 Gettext/Tools

Přidává třídy, které umožní jednodušší práci s shell nástroji pro extrakci řetězců ze zdrojového kódu. Umožňuje specifikovat výstupní formát pro soubory. V aplikaci se používají výhradně soubory POT a PO.

6.1.7 Net/HTTP

Knihovna pro práci s HTTP requesty.

6.1.8 JSON

Knihovna pro JSON formát. Umožňuje generování a parsování JSON formátu.

6.1.9 Base64

Knihovna pro práci s Base64 kódováním.

6.2 Popis hlavních metod Condor gemu

6.2.1 metoda initialize

Metoda která slouží pro inicializaci celého gemu. Jejím parametrem je cesta do složky, kde se nachází konfigurační YAML soubor. Pokud se nepodaří načíst konfigurační soubor, je vyhozena chyba NoConfigFoundError.

Pokud je nalezen konfigurační soubor, je nastavena proměnná config, která je potom dostupná napříč celým gemem. Dále je nastavena proměnná locales_path která značí cestu, kde je inicializovaný náš gem. Jako poslední je volána metoda set_fast_gettext která se stará o nastavení FastGetTextu dle našeho konfiguračního souboru.

Listing 2 condor.rb:self.initialize

```
def self.initialize(locales_path)
  config_path = File.absolute_path('config.yaml',
     locales_path)
  File.exist?(config_path) || raise(NoConfigFoundError,
     config_path)

  @config = YAML.load_file(config_path)['gettext']
  @locales_path = locales_path

  set_fast_gettext
end
```

6.2.2 metoda locale

Stará se o zobrazení aktuální locale tzn. nastavení aktuálně používaného jazyka. Pokud se nepodaří načíst aktuálně používaný jazyk, je načten základní jazyk projektu.

Listing 3 condor.rb:self.locale

```
def self.locale
  FastGettext.locale || default_locale
end
```

6.2.3 metoda locale=

Takzvaná set metoda pro locale metodu. Stará se o nastavení aktuálně používaného jazyka.

Listing 4 condor.rb:self.locale

```
def self.locale=(locale)
  if available_locales.include? locale
    FastGettext.locale = locale
  else
    puts "invalid locale, default locale was set!"
    FastGettext.locale = default_locale
  end
end
```

6.2.4 metoda default_locale

Metoda, která zobrazí základní jazyk projektu z konfiguračního souboru. Pokud není nastaven, je automaticky načtena angličtina.

Listing 5 condor.rb:self.default_locale

```
def self.default_locale
  config['default_locale'] || "en"
end
```

6.2.5 metoda default_locale=

Set metoda pro default_locale metodu. Stará se o nastavení nového základního jazyka pokud je potřeba.

Listing 6 condor.rb:self.default_locale

```
def self.default_locale=(new_locale)
  FastGettext.default_locale = new_locale
  config['default_locale'] = new_locale
end
```

6.2.6 metoda available_locales

Metoda pro výpis povolených jazyků v projektu. Je nastavován z konfiguračního souboru. Pokud je nalezen, je jazyk zvalidován přes seznam všech jazyků z gemu LanguageList. Pokud je jazyk nevalidní, je vyloučen. Pokud se nepodaří načíst z konfiguračního souboru, je vyhozena chyba NoAvailableLocalesFoundError.

Listing 7 condor.rb:self.available_locales

```
def self.available_locales
  if config['available_locales'].nil? || config['
     available_locales'].empty?
    raise NoAvailableLocalesFoundError
  else
    config['available_locales'].reject {|locale| locale.
       empty? || LanguageList::LanguageInfo.
       find_by_iso_639_1(locale).nil?}
  end
end
```

6.2.7 metoda files_to_translate

Metoda, která vypisuje seznam překladatelných souborů. Ten je vytvořen z cest či souborů specifikovaných v konfiguračním souboru. Cesty jsou procházeny rekurzivně a jsou v nich vyhledávány soubory typu .rb(Ruby), .erb, .slim, .haml(šablonovací systémy). Nevalidní cesty či soubory jsou vyloučeny.

Listing 8 condor.rb:self.files_to_translate

```
def self.files_to_translate
  files = (config['source_files'] || []).map do |path|
    Dir.glob(path + "/**/*.{rb,erb,haml,slim}").select{ |
       item| File.file? item}
  end.flatten
  exclusions = (config['exclude_files'] || []).map do |p|
    Dir.glob(p)
  end.flatten
  (files - exclusions)
end
```

6.2.8 metoda pot_file_path

Metoda vracející cestu k souboru typu POT, který slouží jako šablona pro soubory.

Listing 9 condor.rb:self.pot_file_path

```
def self.pot_file_path
  return if locales_path.nil? || config['project_name'].nil
     ?
  File.join(locales_path, config['project_name'] + '.pot')
end
```

6.2.9 metoda po_file_path

Metoda vracející cestu k souboru typu PO. Parametr locale značí jazyk, pro který chceme vrátit PO soubor.

Listing 10 condor.rb:self.po_file_path

```
def self.po_file_path(locale)
  return if locales_path.nil? || config['project_name'].nil
     ? || locale.nil?
  File.join(locales_path, locale, config['project_name'] +
     '.po')
end
```

6.2.10 metoda generate_new_pot

Metoda, která se stará o vygenerování nového POT souboru. Parametr stanoví cestu, kde se má vygenerovaný pot soubor vytvořit. Název je odvozen z názvu projektu v konfiguračním souboru, stejně jako ostatní nastavení. K vygenerování se používají knihovny z GetText/Tools. Konkrétně nástroj XGetText který je určený k práci s POT soubory.

Listing 11 condor.rb:self.generate_new_pot

```
def self.generate_new_pot(path = pot_file_path)
  package_name = config['package_name']
  bugs_address = config['bugs_address']
  copyright_holder = config['copyright_holder']
  comments_tag = config.key?('comments_tag') ? config['
     comments_tag'] : 'TRANSLATORS'
  comments = comments_tag == '' ? '' : '=' + comments_tag
  version = "version"

  xgettext = GetText::Tools::XGetText.new
  xgettext.run("--output", path,"--no-wrap", "--sort-by-
     file", "--msgid-bugs-address", bugs_address, "--
     package-name", package_name, "--package-version",
     version, "--copyright-holder", copyright_holder, "--
     copyright-year", Time.now.year.to_s, "--output-
     encoding", "UTF-8", *files_to_translate)
  if File.exist?(path)
    puts "POT file #{path} created"
    true
  else
    puts 'POT file creation failed'
    false
  end
end
```

6.2.11 metoda generate_new_po

Metoda, která se stará o vygenerování nového PO souboru. Parametr stanoví cestu, kde se má vygenerovat. Druhý parametr říká, pro jaký jazyk se má PO soubor vygenerovat. K práci s PO soubory se používá nástroj MsgInit, pokud je třeba generovat nový, popř. MsgMerge pokud již soubor existuje. MsgMerge se stará o sloučení dvou PO souborů, nového a starého.

Listing 12 condor.rb:self.generate_new_po

```
def self.generate_new_po(language, pot_file = pot_file_path
    )
  language ||= default_locale
  po_file ||= po_file_path(language)
  pot_file ||= pot_file_path

  if language.nil?
    puts "You␣need␣to␣specify␣the␣language␣to␣add.␣Either␣'
       LANGUAGE=eo␣rake␣gettext:po'␣or␣'rake␣gettext:po[
       LANGUAGE]'"
    return
  end

  language_path = File.dirname(po_file)
  FileUtils.mkdir_p(language_path)

  if File.exist?(po_file)
    msgmerge = GetText::Tools::MsgMerge.new
    if msgmerge.run("--update", po_file, pot_file)
      puts "PO␣file␣#{po_file}␣merged"
      true
    else
      puts 'PO␣file␣merge␣failed'
      false
    end
  else
    msginit = GetText::Tools::MsgInit.new
    msginit.run("--input", pot_file, "--output", po_file, "
       --locale", language, "--no-translator")
    if File.exist?(po_file)
      puts "PO␣file␣#{po_file}␣created"
      true
    else
      puts 'PO␣file␣creation␣failed'
      false
    end
  end
end
```

6.2.12 metoda generate_new_pos

Pomocná metoda, která generuje PO soubory pro všechny nastavené překladatelné jazyky.

Listing 13 condor.rb:generate_pos.sync

```
def self.generate_pos
  if available_locales.present?
    available_locales.each do |locale|
      generate_new_po(locale)
    end
  else
    raise(NoConfigFoundError, config_path)
  end
end
```

6.2.13 metoda sync

Metoda, která se stará o synchronizaci klienta se serverovou částí. Její úlohou je vygenerovat nový POT soubor a následně vygenerovat PO soubory pro všechny nastavené jazyky. Poté každý soubor zakóduje pomocí Base64 a odešle na serverovou část aplikace. Po odeslání očekává odpověď v podobě aktuálních nově vygenerovaných PO souborů, které se vygererovaly na straně serveru a uloží je. Zároveň se na server nahrají nové řetězce k překladu.

Listing 14 condor.rb:self.sync

```
def self.sync
  api_key = config['api_key'] || raise(NoApiKeyFoundError)
  uri = URI('https://condor-app.herokuapp.com/api/
     connections/synchronize')
  update_pot

  available_locales.each do |available_locale|
    generate_new_po(available_locale, pot_file_path)

    po_file ||= po_file_path(available_locale)
    file = File.read(po_file)
    encoded_file = Base64.encode64(file)

    res = Net::HTTP.post_form(uri, api_key: api_key, file:
       encoded_file, locale: available_locale)
    jres = JSON.parse(res.body)

    decoded_file = Base64.decode64(jres["file"]).
       force_encoding('UTF-8')
    File.open(po_file, "w:UTF-8:UTF-8") { |po_file| po_file
       .write(decoded_file) }

    initialize(locales_path)
  end
end
```

6.3 Postup při integraci do projektu Hlídačky.cz

Do složky s konfiguračními config/locales je potřeba přidat soubor config.yaml. Ten přidá základní nastavení.

```
---
# This is the project-specific configuration file for setting up
# fast_gettext for your project.
gettext:
  # This is used for the name of the .pot and .po files; they will be
  # called <project_name>.pot?
  project_name: 'hlidacky'
  # This is used in comments in the .pot and .po files to indicate what
  # project the files belong to and should bea little more desctiptive than
  # <project_name>
  package_name: Hlídačky.cz translations
  # api_key for condor-app
  api_key: API_KEY_GOES_HERE
  # The email used for sending bug reports.
  bugs_address: ondrej@zadnik.cz
  # The holder of the copyright.
  copyright_holder: Diploma thesis
  # The locale that the default messages in the .pot file are in
  default_locale: cs
  # Any comments that start with this string will be externalized. (Leave
  # empty to include all.)
  available_locales: [en, cs, sk]
  comments_tag: TRANSLATORS
  # Patterns for +Dir.glob+ used to find all files that might contain
  # translatable content, relative to the project root directory
  source_files:
    - 'app'
```

Obr. 6.2 Soubor config.yaml

Jednotlivé parametry konfiguračního souboru:

- project_name - Název projektu
- package_name - Název balíku který je použitý pro generování POT a PO souborů
- api_key - API klíč který slouží jako autorizační klíč pro serverovou část
- bugs_address - Emailová adresa pro chyby, opět pro POT a PO soubory
- copyright_holder - Držitel práv, slouží jako parametr pro generování POT a PO souborů
- default_locale - Nastavení základního jazyka
- availaible_locales - Pole obsahující překladatelné jazyky
- comments_tag - Překladatelé, další z parametrl pro generování POT a PO souborů

- source_files - Cesta či soubory ve kterých bude gem hledat překladatelné řetězce

Po uložení konfiguračního souboru je potřeba inicializovat samotný Condor gem. Učiníme tak tím že v adresáři naší rails aplikace ve složce initializers, vytvoříme soubot condor.rb kde inicizujeme condor gem s parametrem cesty do složky s konfiguračním souborem.

Listing 15 condor.rb

```
Condor.initialize('config/locales')
```

6.4 Popis Rake úloh Condor gemu

Condor gem poskytuje 3 základní Rake úlohy pro jednodušší práci s gemem. Úlohu pro obnovení POT souboru. Další úlohu, která obnoví PO soubory pro všechny zadané jazyky. Poslední úlohou je synchronizační úloha.

Listing 16 task: generate_new_pot

```
desc "Generate new pot file"
  task :refresh_pot do
    Condor.initialize('config/locales')
    begin
      result = Condor.generate_new_pot
      if result
        puts "POT file #{Condor.pot_file_path} has been
           generated"
      else
        exit 1
      end
    rescue Condor::NoConfigFoundError => e
      puts e.message
    end
  end
```

Listing 17 task: refresh_pos

```
desc "Generate po files for all locales"
  task :refresh_pos do
    Condor.initialize('config/locales')
    begin
      result = Condor.generate_pos
      if result
        puts "PO files for #{Condor.available_locales.to_s}
            has been generated"
      else
        exit 1
      end
    rescue Condor::NoConfigFoundError => e
      puts e.message
    end
  end
```

Listing 18 task: sync

```
desc "Sync all po files with web-app"
  task :sync do
    Condor.initialize('config/locales')
    begin
      result = Condor.sync
      if result
        puts "PO files for #{Condor.available_locales.to_s}
            has been succesfully synced with web app"
      else
        exit 1
      end
    rescue Condor::NoConfigFoundError => e
      puts e.message
    end
  end
```

Tyto úlohy jsou nyní volatelná z konzole pomocí příkazů:

- rake condor:refresh_pot
- rake condor:refresh_pos
- rake condor:sync

Aby bylo možné tyto příkazy volat z Rails aplikace, je nutné je připojit. K tomu slouží třída Railtie. Pomocí Railtie učiníme naše vytvořené Rake úlohy volatelné pomocí dříve zmíněných příkazů v libovolné Rails aplikaci, která použije tento gem.

Listing 19 railtie.rb

```
require 'condor'
require 'rails'

module Condor
  class Railtie < Rails::Railtie
    railtie_name :condor

    rake_tasks do
      spec = Gem::Specification.find_by_name 'condor'
      load "#{spec.gem_dir}/lib/condor/tasks.rake"
    end
  end
end
```

6.5 Testy - Condor

Na Condor gem bylo napsáno několik Unit testů, pro lepší práci při vývoji. Testy jsem ocenil hlavně při změnách či refactoringu kódu.

Třída condor_test.rb:

- test_initialize - test na načtění konfiguračního souboru do proměnno config
- test_set_locale - test na správné nastavení aktualního jazyka
- test_set_invalid_locale - test na nastavení jazyka mimo povolený seznam
- test_defaul_locale - test na zobrazení základního jazyka
- test_available_locales - test na správně nastavené pole jazyků
- test_empty_available_locales - test na vyhození výjimky při prázdném poli jazyků
- test_base_generate_new_pot - test na správné vygenerování nového POT souboru
- test_base_generate_new_po - test na správné vygenerování nového PO souboru

6.6 Serverová část - Condor-app

Druhá část celé aplikace, je webová služba, založená na Ruby on Rails. Ta slouží k registraci uživatele, nastavení projektu který chce přeložit a natáhnutí všech překladatelných řetězců. Po nastavení projektu je vygenerován speciální API klíč, který se zadá do již zmíněné první části. Tím vznikne bezpečné propojení obou aplikací. Poté stačí již jen v aplikaci samotné zavolat speciální příkaz a ten pošle do aplikace extrahované všechny GetText označené řetězce a umožní jejich překlad. Zároveň s tímto se i z aplikace do gemu pošlou přeložené řetězce a vygenerují nové POT a PO soubory.

6.6.1 Databáze

Zde uvádím výpis tabulek

Tabulka uživatelů (Tab. 6.1).

Tab. 6.1 tabulka Users

Identifikátor	Datový typ
id	integer
email	string
name	string
crypted_password	string
password_salt	string
persistence_token	string
single_access_token	string
perishable_token	string
login$_c$*ount*	integer
failed_login_count	integer
last_request_at	datetime
current_login_at	datetime
last_login_at	datetime
current_login_ip	string
last_login_ip	string
deleted_at	datetime
created_at	datetime
updated_at	datetime

Tab. 6.2 tabulka Projects

Identifikátor	Datový typ
id	integer
name	string
description	text
api_key	string
available_locales	string/pqy$_{a}rray$
default_locale	string
deleted_at	datetime
created_at	datetime
updated_at	datetime

Tabulka projektů (Tab. 6.2).

Tab. 6.3 tabulka Collaborators

Identifikátor	Datový typ
id	integer
user_id	integer
project_id	integer
role	string
deleted_at	datetime
created_at	datetime
updated_at	datetime

Tabulka spolupracovníků (Tab. 6.3).

Tab. 6.4 tabulka TranslationKeys

Identifikátor	Datový typ
id	integer
key	string
project_id	integer
source	string
deleted_at	datetime
created_at	datetime
updated_at	datetime

Tabulka Překladových klíčů (Tab. 6.4).

Tab. 6.5 tabulka TranslationTexts

Identifikátor	Datový typ
text	text
locale	string
translation_key_id	integer
deleted_at	datetime
created_at	datetime
updated_at	datetime

Tabulka Překladových textů (Tab. 6.5).

6.6.2 Modely

Popis základních modelů aplikace.

UserSession Model který se stará o přihlašování. Tento model dědí z Authlogic. Je to tedy speciální model, který obsahuje spoustu pomocných metod k přihlašování a přidělování session.

User Model starající se o správu uživatelů. Umožňuje vytvářet nové uživatele a upravovat stávající. Validuje jméno a email. Model uživatelů má vazbu na model spolupracovníků a projektů

Project Model projektů který se stará o vytváření, upravování a správu projektů. Validuje jméno, základní jazyk a dostupné jazyky. Má vazbu na uživatele, spolupracovníky a překladové klíče. Dokáže také přijímat atributy pro překladové klíče.

Collaborator Model který symbolizuje spojovací tabulku mezi uživateli a projekty.

TranslationKey Model překladového klíče. Představuje extrahovaný klíč GetTextu, podle kterého se identifikuje řetězec. Má vazbu na jeden projekt a na překladové texty pro které dokáže přijímat atributy. Validuje klíč a unikátnost páru klíč/zdroj.

TranslationText Model překladových textů. Má vazbu na překladový klíč a validuje jazyk. Tento model symbolizuje jednotlivé překladové texty v různých jazycích.

6.6.3 Kontrolery

Popis základních kontrolerů aplikace.

UserSessionsController Tento kontroler se stará o vytvoření či zničení session pro uživatele. Hlídá také zda již uživatel nějakou session nemá, čili není přihlášen.

UsersController Kontroler pro správu uživatelů. Umožňuje vytvoření nových uživatelů.

ConnectionsController Speciální kontroler zanořený v jmenném prostoru API, je zde akce která hlídá správnost API klíče. Poté zde máme hlavní metodu a to je synchronizační metoda. Přijímá zakódovaný soubor, aktuální jazyk a API klíč. Podle API klíče je vyhledán správný projekt. Soubor je dále dekódován a synchronizován s databází. Poté je vygenerován nový, aktuální soubor PO, který je odeslán jako odpověď do klientské části. Ta jej potom přijme stejným způsobem a získá tak aktuální překladové data.

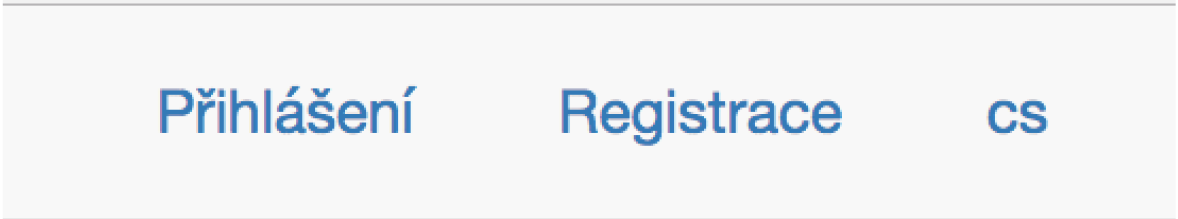
Model TranslationKey Model překladového klíče. Představuje extrahovaný klíč GetTextu podle kterého se identifikuje řetězec. Má vazbu na jeden projekt a na překladové texty pro které dokáže přijímat atributy. Validuje klíč a unikátnost páru klíč/zdroj.

Model TranslationText Model překladových textů. Má vazbu na překladový klíč a validuje jazyk. Tento model symbolizuje jednotlivé překladové texty v různýcj jazycích.

6.6.4 Uživatelské rozhraní

Pro uživatelské rozhraní byl zvolen světlý design, známý jako sb-admin 2. Tato šablona je zdarma pod volně dostupnou licencí. Uživatelské rozhraní se skládá z několika jednoduchých pohledů a kroků.

Úvodní strana Jako první se uživatel setká se stránkou pro přihlášení. Pro jakoukoliv interakci je potřeba používat uživatelský účet. Pro vytvoření uživatelského účtu je potřeba kliknout vpravo nahoře do menu na odkaz registrace. Pokud již účet má, stačí se na hlavní stránce přihlásit.

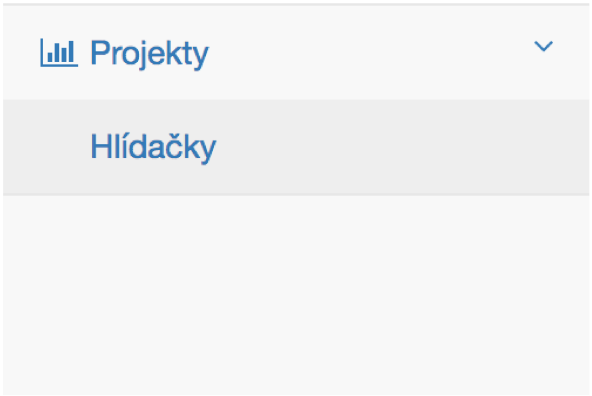


Obr. 6.3 Inline menu pro přihlášení, registraci a změnu jazyka

Obr. 6.4 Formulář pro přihlášení

Obr. 6.5 Formulář pro registraci

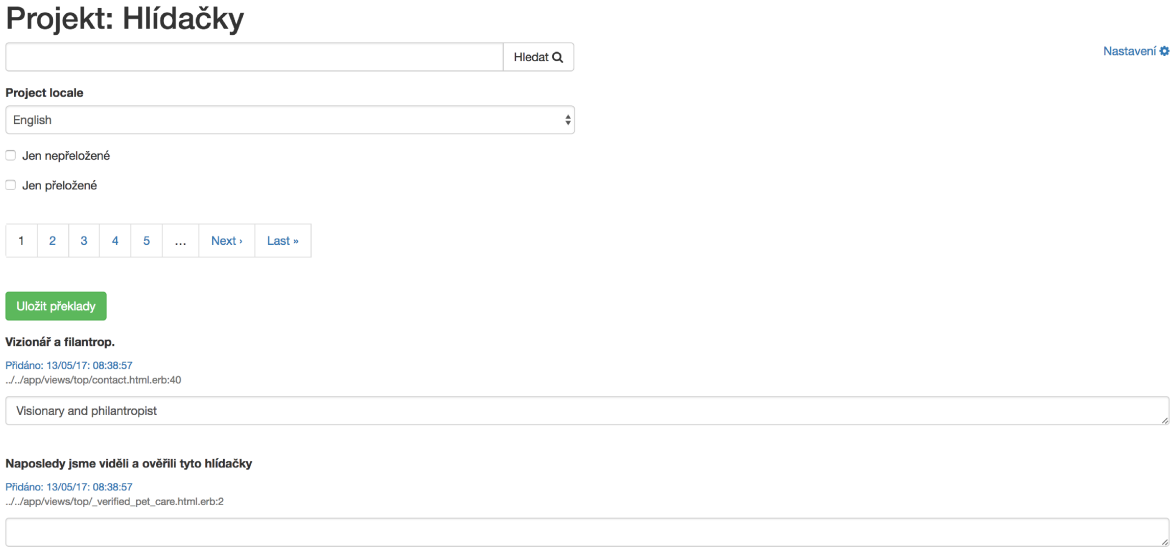
Hlavní strana Po přihlášení se dostaneme na hlavní stranu celé aplikace. Pokud již máme vytvořený nějaký projekt, vidíme jej na levé straně ve sloupci našich projektů. Pokud ne, uživatel je přesměrován na vytvoření nového projektu. Po vytvoření se dostaneme zpět a vidíme jej v levém sloupci.



Obr. 6.6 Levé menu projektů

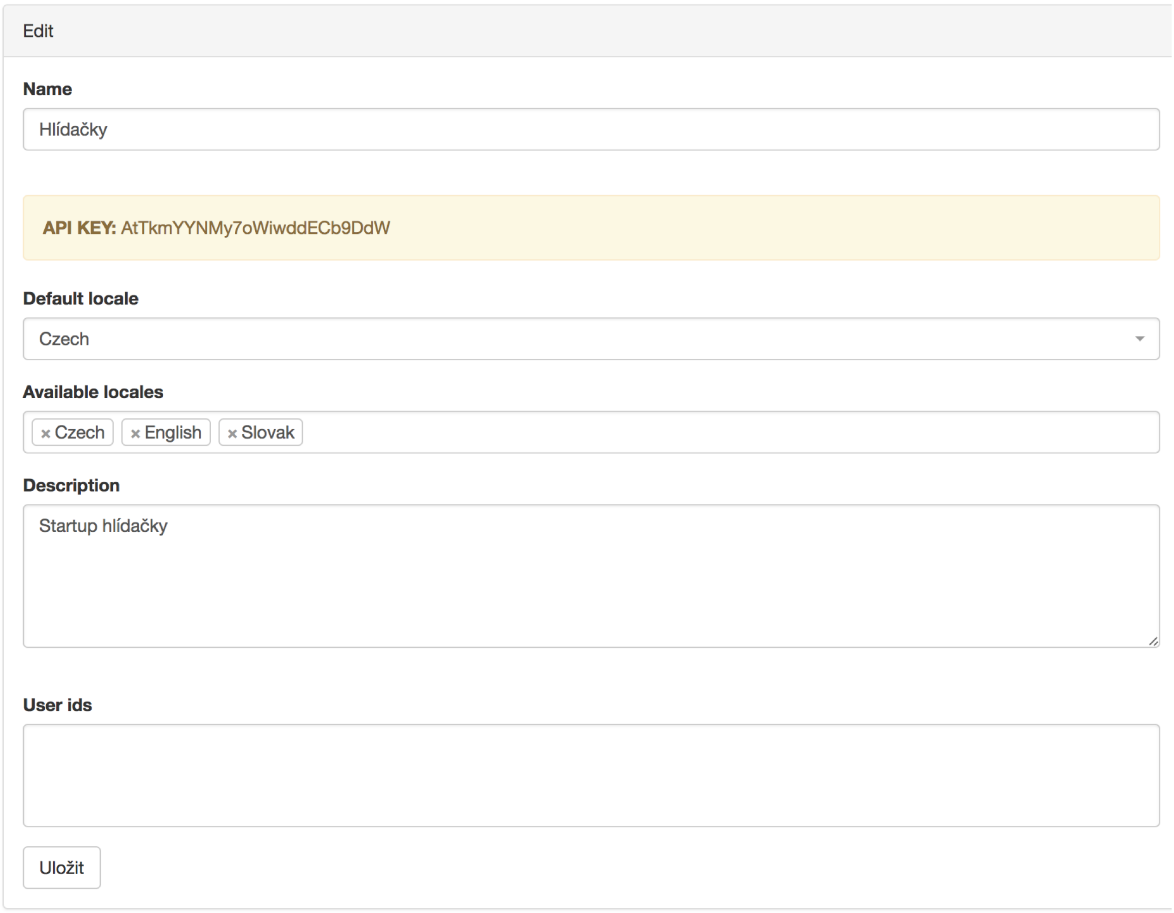
Překlady Nyní se dostáváme k srdci aplikace, tím je výpis překladů. Nahoře vidíme přehledně vypsaný název Projektu. Pod ním najdeme vyhledávací pole. Vyhledávání funguje jako full-text. Dokáže vyhledávat dle celých slov nebo jen jejich částí. Jako další

najdeme přepínač projektových jazyků. Vždy jsou zobrazeny či ukládány překlady pro aktuálně zvolený jazyk. Dále jsou k dispozici filtry, zaškrtnout můžeme přeložené či nepřeložené texty. Jako poslední je zde jednoduché stránkování.



Obr. 6.7 Hlavní stránka s překlady

Nastavení projektu Ještě bych zde rád zmínil nastavení projektu, do toho se dostaneme přes tlačítko které je umístěné v pravém horním rohu. Najdeme zde jednoduchý formulář, jako při vytváření nového s tím rozdílem, že je zde umístěn vygenerovaný API klíč. Ten je potřebný k autorizaci při synchronizaci.



Obr. 6.8 Nastavení projektu

6.7 O projektu Hlídačky.cz

Hlídačky.cz jsou startupový projekt z České republiky. Je to webová služba, kde se může zaregistrovat jakýkoliv člověk co hledá nebo nabízí hlídání dětí, úklid nebo hlídání domácích zvířat. Tyto dvě skupiny jsou potom propojeny v nabídku a poptávku jednotlivých služeb. Můžou spolu komunikovat prostřednictvím zpráv nebo si jen vzít telefonní číslo. Je dbán zřetel na to aby vše bylo co nejjednodušší a nejrychlejší.

Skupina hlídaček, jak se zde říká slečnám poskytujícím hlídání či úklid, má možnost několika stupňů ověření a širokých možností vyplnění profilu. Proto si zde většina rodičů dokáže pomocí rozsáhlého filtrování najít to co potřebují. Později mohou psát reference na jednotlivé slečny dle jejich spokojenosti či nespokojenosti. Registrace je zde pro všechny zdarma.

6.8 Integrace Condoru do projektu Hlídačky.cz

Hlídačky.cz jsou postaveny na frameworku Ruby on Rails verze 5. Proto jsem zvolil i já pro celou aplikaci jazyk Ruby a framework Ruby on Rails, kvůli snazší integraci. Condor gem stačí přidat do Gemfilu webové aplikace Hlídačky.cz.

Poté přidáme inicializer, standartně tak jak jsem popisoval integraci do projektu výše. Pokud máme správně přidaný Condor gem v Gemfilu a přidaný inicializer, přidáme ještě konfigurační soubor config.yaml do složky odkud budeme chtít načítat.

Předtím si ještě vytvoříme účet na Condor-app, kde si vytvoříme projekt a nastavíme ho stejně jako náš konfigurační soubor. Po vytvoření projektu získáme API klíč, ten si uložíme taktéž do konfiguračního souboru.

Pokud jsme všechny kroky provedli správně, máme tedy kompletně nastavený Condor gem a Condor-app. Můžeme použít příkaz rake condor:sync. Tím byly synchronizovány klíče a překlady. Ve webové aplikace nyní vidíme natažené všechny GetTextové řetězce a jejich překlady pro konkrétní jazyky.

Při každé další změně, stačí již jen používat rake condor:sync. Ten nám zajistí aktuálnost jak klientské části tak serverové. Aplikace Hlídačky.cz tedy získala funkčnost, kdy programátorovi stačí jedním příkazem synchronizovat svoji aplikaci. O překlady se může starat kdokoliv.

6.8.1 Proces překladů před nasazením aplikace Condor

Jak jsem zmínil na začátku aktuální systém pro lokalizaci na Hlídačky.cz je Translation IO. Proces překladů je víceméně stejný jako u implementované aplikace, bohužel webové UI je nepřehledné a matoucí. Taktéž cena pro aktuální počet klíčů na Hlídačky.cz je dlouhodobě neúnosná a bude stále stoupat s přibývajícími texty.

6.8.2 Proces překladů po nasazení aplikace Condor

Implementovaná aplikace plně splňuje požadavky na funkčnost, použité technologie. Dále je vyhovující z hlediska uživatelského rozhraní a postupu při překládání a synchronizaci.

Do budoucna bych aplikaci rád obohatil o možnost přímého ukládání do databáze. Tohle by mohlo přinést jedinečnou funkcionalitu, kdy by bylo možné updatovat překlady za chodu, bez nutnosti synchronizace souborů. Dále bych rád implementoval našeptávač překladů který dokáže rozpoznat z jakého jazyka překládat a navrhnout překlad.

Určitě bych se rád věnoval vylepšování této aplikace dále, věřím že je to přínosný nástroj, který ocení více podobných začínajících projektů.

ZÁVĚR

Cílem této práce bylo shrnout aktuální možnosti překladů webových aplikací. Nastudovat aktuální technologie, které by mohly být užitečné v budoucí implementaci. Vyhodnotit správné funkční a nefunkční požadavky. Ty byly vyhodnoceny, při detailním rozboru požadavků se zadavatelem Hlídačky.cz. Klíčovým požadavkem, se zdála být jednoduchá integrace do již zaběhnuté aplikace s minimem změn. Dalším důležitým požadavkem byla jednoduchost a intuitivnost uživatelského rozhraní.

Technologie byly zvoleny s ohledem na programovací jazyk a framework celého startupu Hlídačky.cz. Všechny vybrané technologie, byly zvoleny s ohledem na možný růst aplikace a její rozšiřování v budoucnu. Výběr potřebných technologií byl odsouhlasen zadavatelem. Všechny technologie které byly použity, jsou totožné s technologiemi a knihovnami používanými zadavatelem. Velkým plusem je tedy rychlá adaptace programátorů a možnost dalšího rozšiřování aplikace.

Implementace samotná, probíhala v souladu s konzultacemi se zadavatelem. Zvolení vývojového frameworku se ukázalo jako vhodné. Webová aplikace by měla být schopná optimálního zobrazení i na mobilních zařízeních a tabletech. Pro zlepšení stability a pomoc při vývoji byly použity unit a controller testy. Ty se ukázaly jako výborná pomoc při vývoji.

Přínosem celé aplikace bylo snížení nákladů pro zadavatele a udržení stávající architektury. Aplikace se tedy dokázala adaptovat, na již používané knihovny a překlady. Dalším splněným bodem je jednoduchá ovladatelnost a totožný postup používání jako minulé řešení.

SEZNAM POUŽITÉ LITERATURY

[1] *METZ, Sandi. Practical object-oriented design in Ruby: an agile primer. Upper Saddle River, NJ: Addison-Wesley, 2013. ISBN 0321721330..*

[2] *BROWN, Gregory T. Ruby best practices. 1st ed. Sebastopol, CA: O'Reilly, c2009. ISBN 0596523009..*

[3] *BY SAM RUBY, Dave Thomas. Agile web development with Rails 4. Dallas, Texas: Pragmatic Bookshelf, 2013. ISBN 9781937785567..*

[4] *OLSEN, Russ. Design patterns in Ruby. Upper Saddle River, NJ: Addison-Wesley, c2008. ISBN 0321490452..*

[5] *ORSINI, Rob. Rails cookbook. Farnham: O'Reilly, 2007. ISBN 0596527314..*

[6] *FULTON, Hal. The Ruby way: solutions and techniques in Ruby programming. 3rd ed. Boston, Mass: Addison-Wesley, 2010. ISBN 0321714636..*

[7] *Ruby on Rails guides* [online]. [cit. 2017-05-13]. Dostupný z WWW: `http://guides.rubyonrails.org//`.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Startup	Nově založená a rychle se vyvíjející společnost.
IT	Informační technologie
YAML	Formát pro serializaci dat textových souborů
Framework	Softwarová struktura, která slouží jako podpora při programování a vývoji a organ
HTML	Značkovací jazyk používaný pro tvorbu webových stránek
XML	Obecný značkovací jazyk
HTTP	Internetový protokol určený pro výměnu hypertextových dokumentů
HSTORE	Postgres datový typ klíč/hodnota
JSON	Způsob zápisu dat nezávislý na počítačové platformě
PaaS	Platforma jako služba
POT	Šablona katalogu
PO	Soubor vyextrahovaných textů
API	Rozhraní pro programování aplikací

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. OBSAH CD

- Diplomová práce ve formátu PDF/A
- Zdrojové kódy LATEX
- Zdrojové kódy aplikace